AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method for operating a multi-threaded system having a plurality of active threads, the method comprising:

assigning a ~~unique~~ interrupt execution priority value to each of a plurality of interrupts;

specifying a global interrupt threshold value that is ~~applicable to all~~higher than the execution priority of each of the plurality of active threads; and

processing~~, by a thread execution logic,~~ a requested interrupt only when the ~~unique~~ interrupt execution priority value assigned to the requested interrupt is higher than the global interrupt threshold value.

2. (Original) The method of claim 1, wherein processing the requested interrupt comprises:

performing an interrupt entry process to prepare for an interrupt service routine (ISR);

executing the ISR; and

performing an interrupt exit process to return control from the ISR.

3. (Original) The method of claim 2, wherein performing the interrupt entry process comprises:

identifying one of the plurality of active threads as an interrupt thread;

switching to the interrupt thread if the interrupt thread is not executing; and

branching to the ISR.

4. (Original) The method of claim 3, wherein each of the plurality of active threads comprises a thread context, and wherein performing the interrupt entry process further comprises saving the thread context of the interrupt thread.

5. (Original) The method of claim 4, wherein performing the interrupt exit process comprises:

executing a return from exception (RFE) instruction; and

restoring the thread context of the interrupt thread.

6. (Original) The method of claim 5, wherein performing the interrupt entry process further comprises disabling interrupts and thread switching.

7. (Original) The method of claim 6, wherein performing the interrupt exit process further comprises enabling interrupts and thread switching.

8. (Original) The method of claim 6, wherein executing the ISR comprises enabling interrupts and thread switching after a predetermined interval.

9. (Original) The method of claim 3, wherein each of the plurality of active threads consists of a first set of context registers and a second set of context registers, wherein performing the

interrupt entry process further comprises saving the first set of context registers of the interrupt thread, and wherein executing the ISR comprises:

saving the second set of context registers of the interrupt thread if the second set of context registers of the interrupt thread are required for servicing the requested interrupt;

servicing the requested interrupt; and

restoring the second set of context registers of the interrupt thread after servicing the requested interrupt if the second set of context registers of the interrupt thread were required for servicing the requested interrupt.

10. (Original) The method of claim 9, wherein performing the interrupt exit process comprises:

executing a return from exception (RFE) instruction; and

restoring the upper context registers of the interrupt thread.

11. (Original) The method of claim 1, further comprising processing traps only in the active threads originating the traps.

12. (Original) The method of claim 11, wherein processing traps comprises:

detecting a trap from an originating thread, the originating thread being one of the plurality of active threads;

storing trap background data for the trap if the trap is asynchronous; and

associating a trap pending indicator with the originating thread if the originating thread is not executing.

13. (Original) The method of claim 12, wherein processing traps further comprises:

performing a trap entry process to prepare for a trap handling routine;

executing the trap handling routine; and

performing a trap exit process to return control from the trap handling routine.

14. (Original) The method of claim 13, wherein each of the plurality of active threads comprises a thread context, and wherein performing the trap entry process comprises:

waiting for originating thread to start executing if the originating thread is not executing;

saving the thread context of the originating thread; and

branching to a trap handler.

15. (Original) The method of claim 14, wherein waiting for the originating thread to start executing comprises monitoring the plurality of active threads until execution switches to one of the plurality of active threads associated with the trap pending indicator.

16. (Original) The method of claim 14, wherein performing the trap exit process comprises:

executing a return from trap instruction; and

restoring the context of the originating thread.

17. (Original) The method of claim 16, wherein performing the trap entry process further comprises disabling interrupts and thread switching.

18. (Original) The method of claim 17, wherein performing the trap exit process further comprises enabling interrupts and thread switching.

19. (Original) The method of claim 17, wherein executing the trap handling routine comprises enabling interrupts and thread switching after a predetermined interval.

20. (Original) The method of claim 13, wherein each of the plurality of active threads consists of a first set of context registers and a second set of context registers, wherein performing the trap entry process further comprises saving the first set of context registers of the originating thread, and wherein executing the trap handling routine comprises:

saving the second set of context registers of the originating thread if the second set of context registers of the originating thread are required for servicing the trap;

servicing the trap; and

restoring the second set of context registers of the originating thread after servicing the trap if the second set of context registers of the interrupt thread were required for servicing the trap.

21. (Original) The method of claim 20, wherein performing the interrupt exit process comprises:

executing a return from trap instruction; and

restoring the upper context registers of the originating thread.

22. (Currently Amended) A method for operating a multi-threaded system having a plurality of active threads, the method comprising:

accepting a request for an interrupt;

switching execution to a predetermined one of the plurality of active threads; and

executing an interrupt service request from the predetermined one of the plurality of active threads to service the interrupt,

wherein accepting a request for an interrupt comprises:

assigning a ~~unique~~ interrupt execution priority value to each interrupt for the multi-threaded embedded system;

specifying a global interrupt threshold value that is ~~applicable to all~~higher than the execution priority of each of the plurality of active threads; and

taking~~, by a thread execution logic,~~ the interrupt only when the ~~unique~~ interrupt execution priority value assigned to the interrupt is higher than the global interrupt threshold value.

23-24. (Canceled)

25. (Currently Amended) A multi-threaded system comprising:

thread execution logic ~~for generating~~configured to generate instruction requests from an executing thread; and

threshold interrupt logic ~~for generating~~configured to generate a global interrupt threshold value that is higher than the execution priority of each~~applicable to all~~ of the threads of the multi-threaded system, wherein the thread execution logic only accepts interrupts assigned a ~~unique~~ interrupt execution priority value higher than the global interrupt threshold value.

26. (Currently Amended) The multi-threaded system of claim 25, further comprising interrupt thread logic ~~for switching~~configured to switch execution to a selected interrupt thread before servicing any interrupt.

27. (Currently Amended) The multi-threaded system of claim 26, further comprising disabling logic ~~for disabling~~configured to disable interrupts and thread switching while an interrupt is being serviced.

28. (Currently Amended) The multi-threaded system of claim 27, further comprising thread tagging logic ~~for storing~~configured to store trap background data for asynchronous traps, wherein every trap is handled in its originating thread.

29. (Currently Amended) A multi-threaded system comprising:

means for specifying a global interrupt threshold value that is higher than the execution priority of eachapplicable to all of the threads of the multi-threaded system; and

thread execution logic means for processing a requested interrupt only when a unique interrupt execution priority value assigned to the requested interrupt is higher than the global interrupt threshold value.

30. (Original) The multi-threaded system of claim 29, wherein the means for processing the requested interrupt comprises:

means for identifying one of the plurality of active threads as an interrupt thread;

means for switching to the interrupt thread if the interrupt thread is not executing; and

means for branching to the ISR.

31. (Original) The multi-threaded system of claim 30, wherein the means for processing the requested interrupt further comprises means for saving a thread context of the interrupt thread.

32. (Original) The multi-threaded system of claim 31, wherein the means for processing the requested interrupt further comprises means for restoring the thread context of the interrupt thread after executing a return from exception (RFE) instruction.

33. (Original) The multi-threaded system of claim 32, wherein the means for processing the requested interrupt further comprises means for disabling interrupts and thread switching during execution of the ISR.

34. (Original) The multi-threaded system of claim 29, further comprising means for processing traps, the means for processing traps comprising:

means for detecting a trap from an originating thread;

means for storing trap background data for the trap if the trap is asynchronous;

means for processing the trap if the originating thread is executing; and

means for associating a trap pending indicator with the originating thread if the originating thread is not executing.

35. (Original) The multi-threaded system of claim 34, wherein the means for processing traps further comprises means for detecting execution of an active thread associated with the trap pending indicator.

36. (Original) The multi-threaded system of claim 35, wherein the means for processing the trap comprises means for disabling interrupts and thread switching.

37. (Previously Presented) The method of claim 1, wherein the global interrupt threshold value is a single global interrupt threshold value.

38. (Previously Presented) The method of claim 22, wherein the global interrupt threshold value is a single global interrupt threshold value.

39. (Previously Presented) The multi-threaded system of claim 25, wherein the global interrupt threshold value is a single global interrupt threshold value.

40. (Previously Presented) The multi-threaded system of claim 29, wherein the global interrupt threshold value is a single global interrupt threshold value.

41. (New) A method for operating a multi-threaded system, the method comprising:

storing trap background data for an asynchronous trap generated from a thread;

generating a trap pending indicator for the thread; and

servicing the asynchronous trap using the stored background data, due to the trap pending indicator, when the thread resumes execution.

42. (New) A multi-threaded system comprising:

trap tagging logic configured to record trap background data for an asynchronous trap, and to generate a trap pending indicator for a thread in which the asynchronous trap occurred; and

thread execution logic configured to service the asynchronous trap using the recorded trap background data, due to the trap pending indicator, when the thread resumes execution.